You can't see me
A Mac OS X Rootkit uses the tricks you haven't known yet

Team T5

Ming-chieh Pan
Sung-ting Tsai

About Us

Team T5

We monitor, analyze, and track cyber threats.

CHROOT

Team T5 Research

Sourcing

Unique Collections

Analysis

Deep Insight of Threats

Product

Intelligence Report

Sung-ting Tsai (TT)

Team T5

Leader

Speech

Black Hat USA 2011 / 2012
Codegate 2012
Syscan 10’ / 12’
HITCon 08’

Research

New security technology
Malicious document
Malware auto-analyzing system (sandbox technologies)
Malware detection
System vulnerability and protection
Mobile security

Ming-chieh Pan (Nanika)

Team T5 Inc.

Chief Researcher

Speech

Black Hat USA 2011 / 2012

Syscan Singapore/Taipei/Hong Kong 08/10

Hacks in Taiwan Conference
05/06/07/09/10/12

Research

Vulnerability discovery and analysis
Exploit techniques
Malware detection
Mobile security

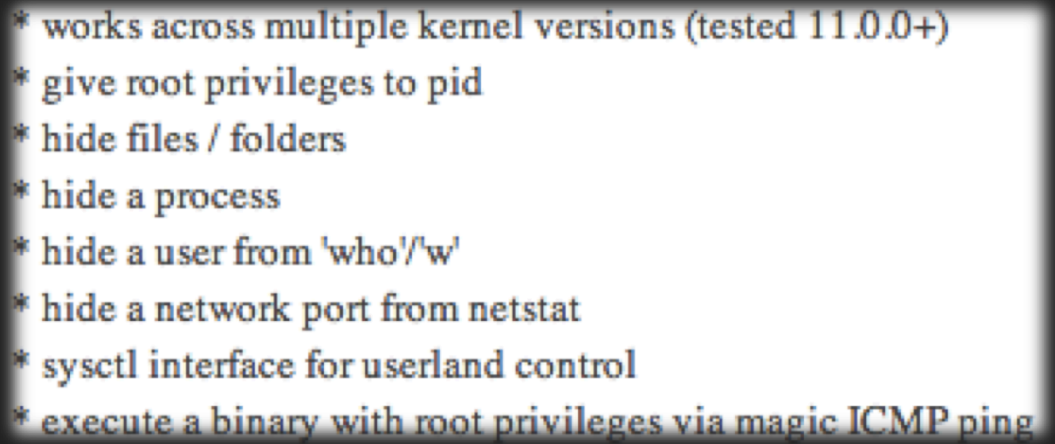
Agenda

- Advanced Process Hiding
- A Privileged Normal User
- Direct Kernel Task Access (Read/Write)
- Loading Kernel Module Without Warnings
- A Trick to Gain Root Permission

Advanced Process Hiding

DKOM
launchd

The rubilyn Rootkit

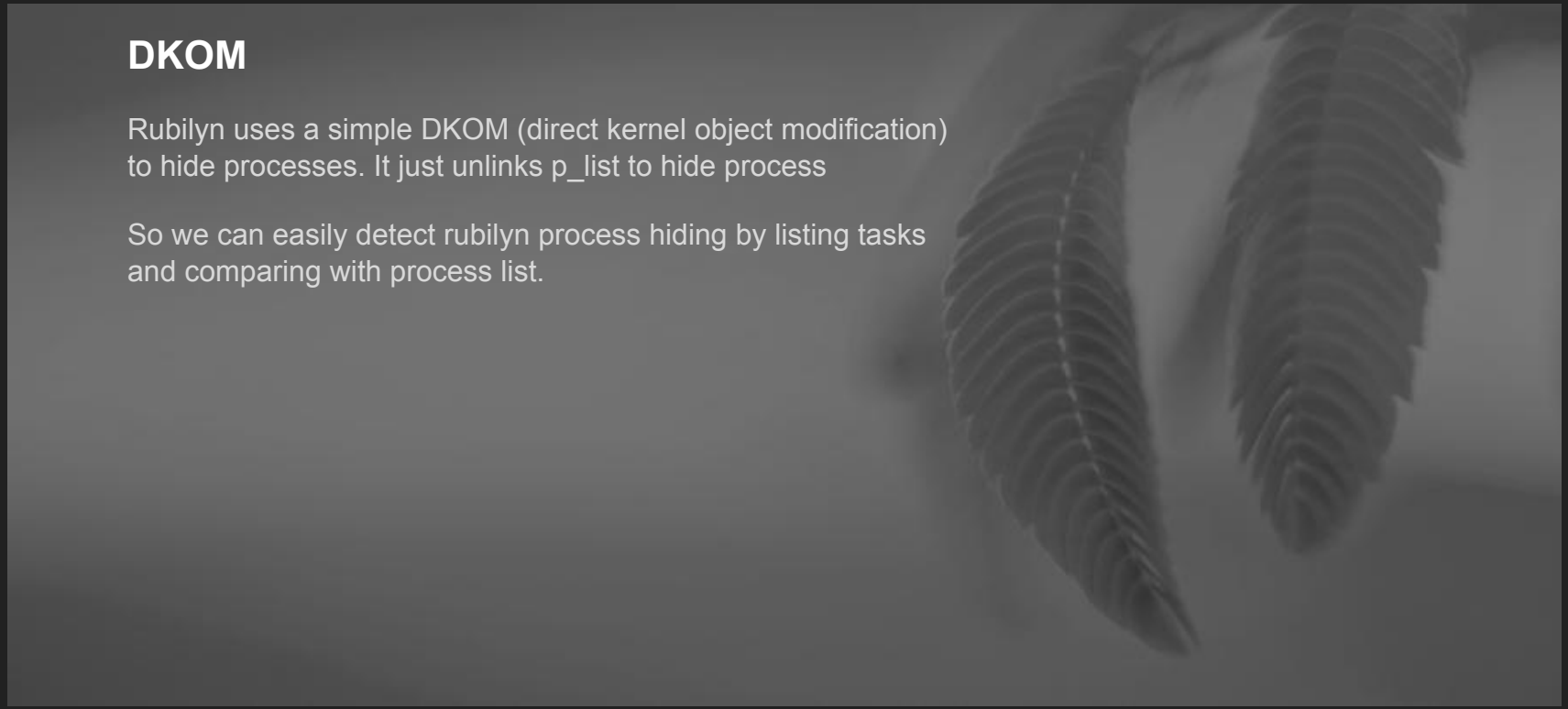


Using DKOM to hide process

Process Structure in Kernel

```
struct  proc {
        LIST_ENTRY(proc) p_list;                /* List of all processes. */

        pid_t           p_pid;                  /* Process identifier. (static)*/
        void *          task;                   /* corresponding task (static)*/
        struct  proc *  p_pptr;                 /* Pointer to parent process.(LL) */
        pid_t           p_ppid;                 /* process's parent pid number */
        pid_t           p_pgrpid;               /* process group id of the process (LL)*/

        lck_mtx_t       p_mlock;                /* mutex lock for proc */

        char            p_stat;                 /* S* process status. (PL)*/
        char            p_shutdownstate;
        char            p_kdebug;               /* P_KDEBUG eq (CC)*/
        char            p_btrace;               /* P_BTRACE eq (CC)*/

        LIST_ENTRY(proc) p_pglist;              /* List of processes in pgrp.(PGL) */
        LIST_ENTRY(proc) p_sibling;             /* List of sibling processes. (LL)*/
        LIST_HEAD(, proc) p_children;           /* Pointer to list of children. (LL)*/
        TAILQ_HEAD( , uthread) p_uthlist;       /* List of uthreads  (PL) */
```

Detecting rubilyn Process Hiding



DKOM

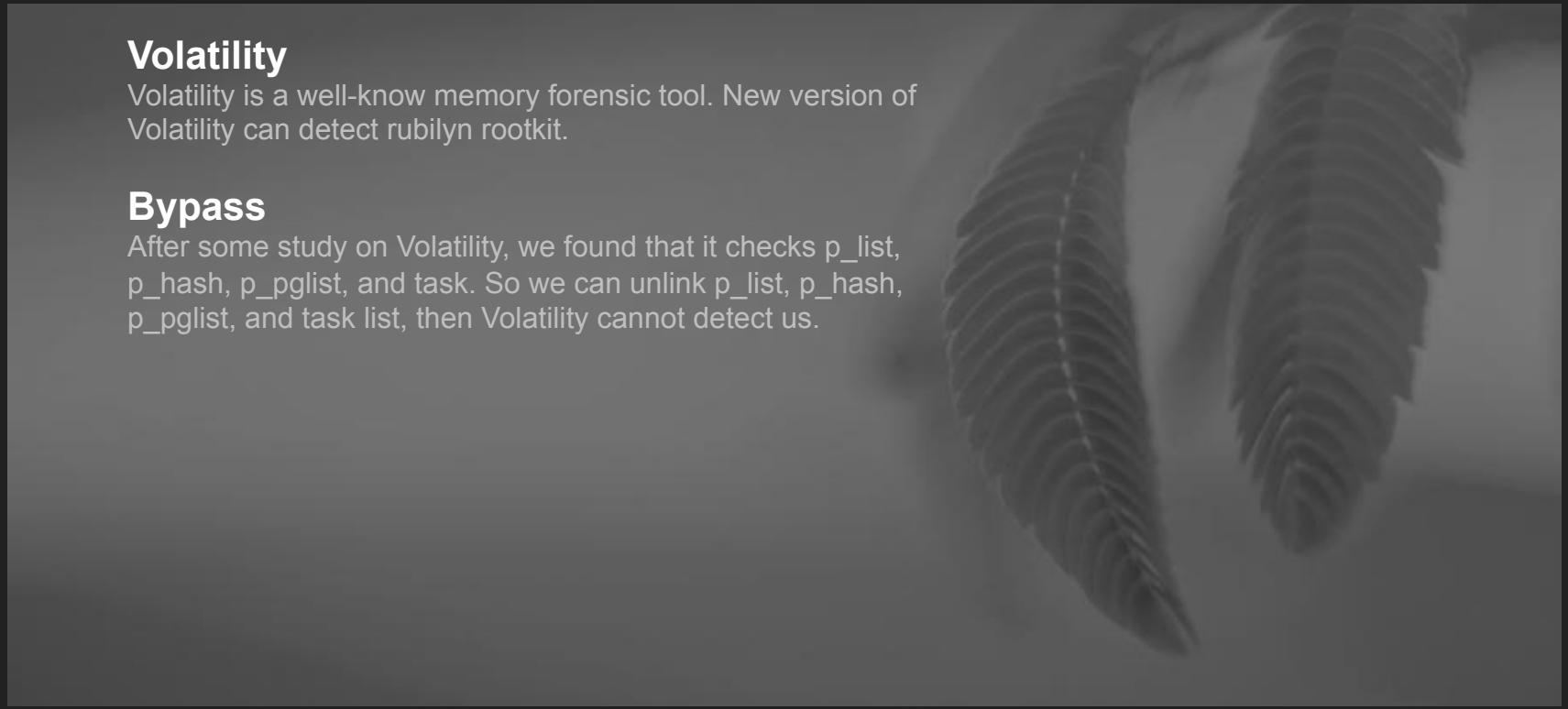
Rubilyn uses a simple DKOM (direct kernel object modification) to hide processes. It just unlinks p_list to hide process

So we can easily detect rubilyn process hiding by listing tasks and comparing with process list.

```
struct  proc {
        LIST_ENTRY(proc) p_list;                /* List of all processes. */

        pid_t           p_pid;                  /* Process identifier. (static)*/
        void *          task;                   /* corresponding task (static)*/
        struct  proc *  p_pptr;                 /* Pointer to parent process.(LL) */
        pid_t           p_ppid;                 /* process's parent pid number */
        pid_t           p_pgrpid;               /* process group id of the process (LL)*/
```

```
struct task {
        /* Synchronization/destruction information */
        decl_lck_mtx_data(,lock)                        /* Task's lock */
        uint32_t        ref_count;      /* Number of references to me */
        boolean_t       active;         /* Task has not been terminated */
        boolean_t       halting;        /* Task is being halted */

        /* Miscellaneous */
        vm_map_t        map;            /* Address space description */
        queue_chain_t   tasks;  /* global list of tasks */
        void            *user_data;     /* Arbitrary data settable via IPC */

        /* Threads in this task */
        queue_head_t            threads;
```

Volatility and Bypass Volatility



Volatility

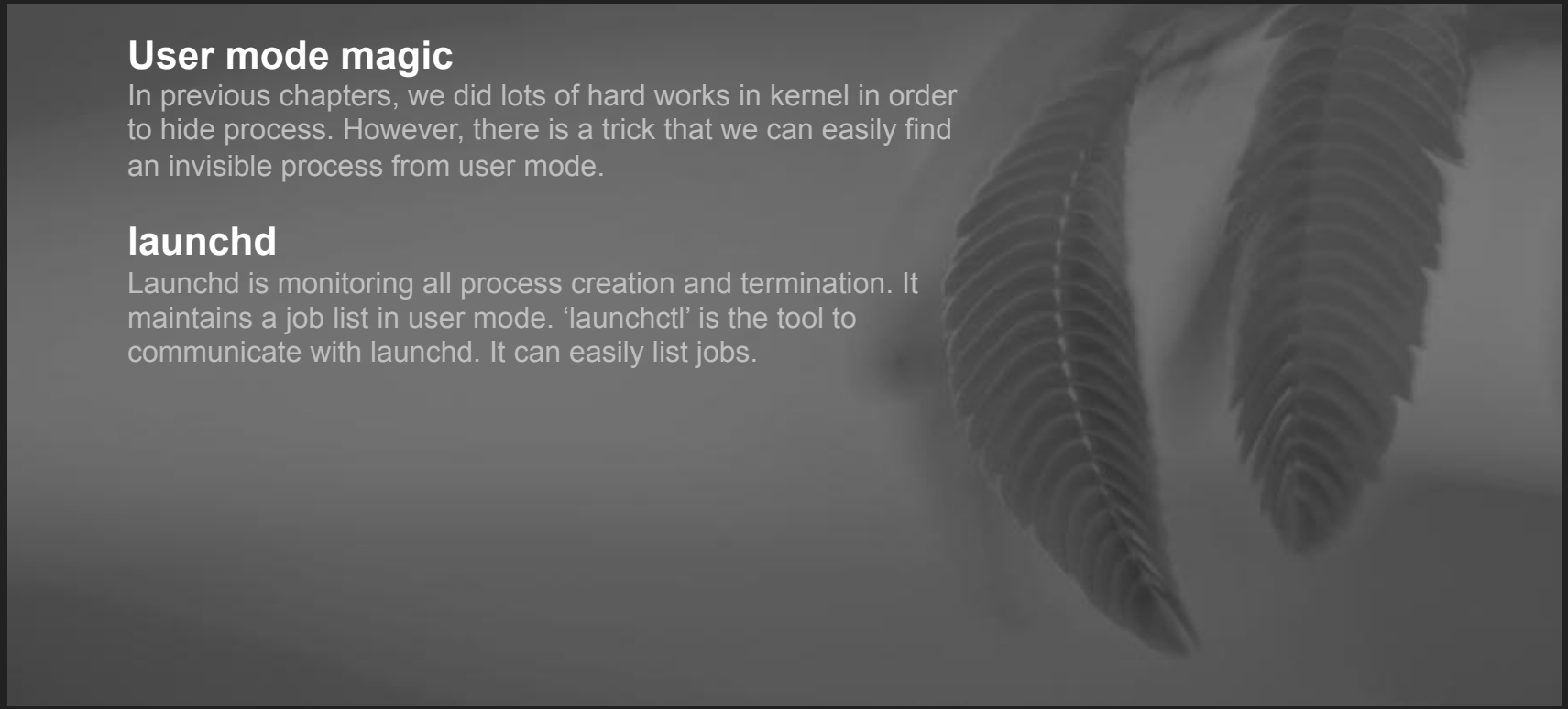
Volatility is a well-know memory forensic tool. New version of Volatility can detect rubilyn rootkit.

Bypass

After some study on Volatility, we found that it checks p_list, p_hash, p_pglist, and task. So we can unlink p_list, p_hash, p_pglist, and task list, then Volatility cannot detect us.


DEMO 0x01
Bypass Volatility

Launchd Magic



User mode magic

In previous chapters, we did lots of hard works in kernel in order to hide process. However, there is a trick that we can easily find an invisible process from user mode.

launchd

Launchd is monitoring all process creation and termination. It maintains a job list in user mode. ‘launchctl’ is the tool to communicate with launchd. It can easily list jobs.

```
Naniteki-MacBook-Air:ext_research Nani$ launchctl list
PID     Status  Label
11665   -       0x7fc8e9c3b1a0.anonymous.launchctl
11648   -       0x7fc8e9d07a00.anonymous.vmware-vmx
11511   -       [0x0-0x5ab5ab].com.SweetScape.010Editor
11483   -       0x7fc8e9e0e9b0.anonymous.Google Chrome H
11401   -       0x7fc8e9c390f0.anonymous.Google Chrome H
11305   -       0x7fc8e9e0c7c0.anonymous.Google Chrome H
11263   -       0x7fc8e9d07700.anonymous.Google Chrome H
11253   -       0x7fc8e9d06d90.anonymous.Google Chrome H
11178   -       0x7fc8e9e0cdc0.anonymous.Google Chrome H
10785   -       0x7fc8e9e0cac0.anonymous.Google Chrome H
10411   -       0x7fc8e9c3b4a0.anonymous.Google Chrome H
10341   -       0x7fc8e9c3aea0.anonymous.Google Chrome H
10312   -       0x7fc8e9d07100.anonymous.Google Chrome H
10237   -       0x7fc8e9c3aba0.anonymous.vmnet-dhcpd
10247   -       0x7fc8e9c3a390.anonymous.vmware-usbarbit
10242   -       0x7fc8e9c3a8a0.anonymous.vmnet-netifup
10240   -       0x7fc8e9c39d90.anonymous.vmnet-natd
```

Unlink a job in Launchd

Get root permission

Enumerate process launchd and get launchd task

Read launchd memory and find data section

Find root_jobmgr
Check root_jobmgr->submgrs and submgrs->parentmgr

Enumerate jobmgr and get job

Enumerate job and find the target job
Information Storage

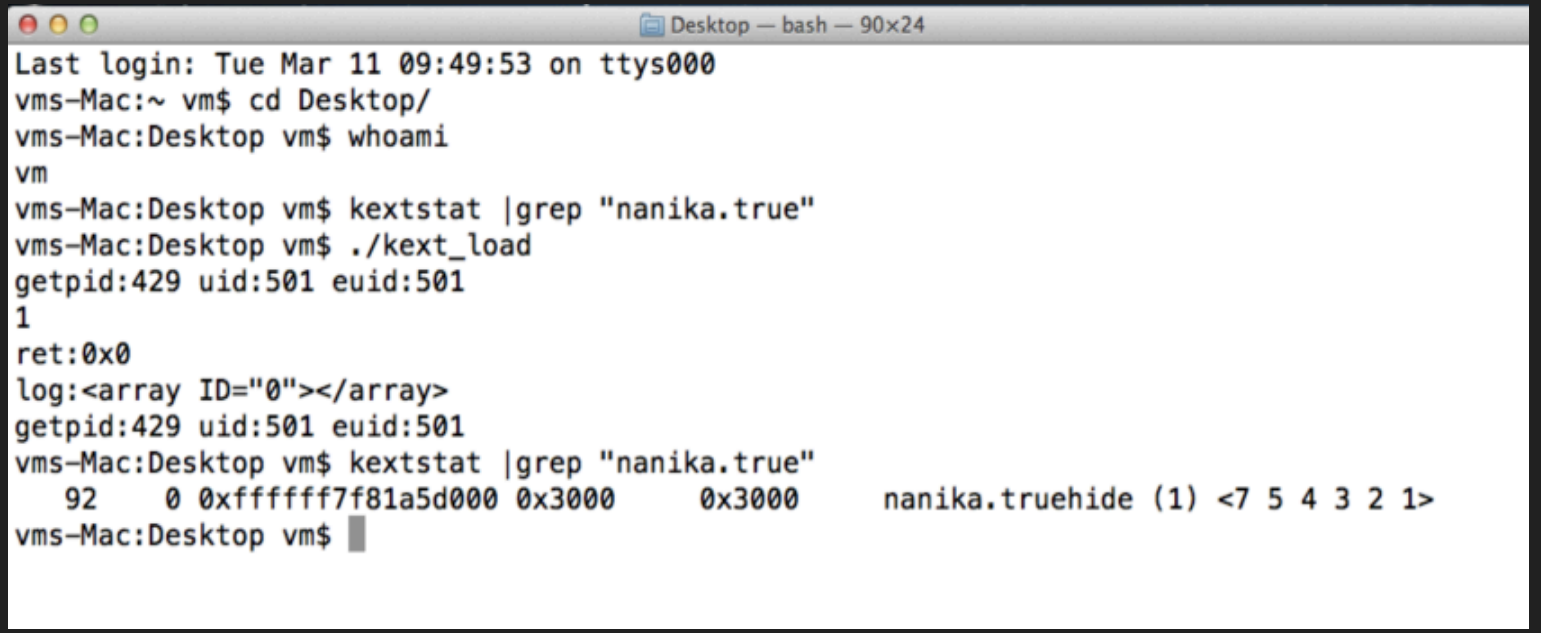
Unlink the job


DEMO 0x02
Remove job from launchd

A Privileged Normal User

host privilege

Running Privileged Tasks as a Normal User



Host Privilege

```
struct  host {
        decl_lck_mtx_data(,lock)                /* lock to protect exceptions */
        ipc_port_t special[HOST_MAX_SPECIAL_PORT + 1];
        struct exception_action exc_actions[EXC_TYPES_COUNT];
};

typedef struct host     host_data_t;

extern host_data_t      realhost;
```

```
/*
 * Always provided by kernel (cannot be set from user-space).
 */
#define HOST_PORT                       1
#define HOST_PRIV_PORT                  2
#define HOST_IO_MASTER_PORT             3
#define HOST_MAX_SPECIAL_KERNEL_PORT    7 /* room to grow */
```

Host Interface

host_get_clock_service - Return a send right to a kernel clock's service port.
host_get_time - Returns the current time as seen by that host.
host_info - Return information about a host.
host_kernel_version - Return kernel version information for a host.
host_statistics - Return statistics for a host.
mach_host_self - Returns send rights to the task's host self port.

Data Structures

host_basic_info - Used to present basic information about a host.
host_load_info - Used to present a host's processor load information.
host_sched_info - - Used to present the set of scheduler limits associated with the host.
kernel_resource_sizes - Used to present the sizes of kernel's major structures.

Host Control Interface

host_adjust_time - Arranges for the time on a specified host to be gradually changed by an adjustment value.
host_default_memory_manager - Set the default memory manager.
host_get_boot_info - Return operator boot information.
host_get_clock_control - Return a send right to a kernel clock's control port.
host_processor_slots - Return a list of numbers that map processor slots to active processors.
host_processors - Return a list of send rights representing all processor ports.
host_reboot - Reboot this host.
host_set_time - Establishes the time on the specified host.

Host Security Interface

host_security_create_task_token - Create a new task with an explicit security token.
host_security_set_task_token - Change the target task's security token.

processor_set_default
host_processor_set_priv
processor_set_tasks

How to Get Host Privilege

Assign host privilege to a task

VParse mach_kernel and find _realhost
Find task structure
Assign permission: task->itk_host = realhost->special[2]
Then the task/process can do privilege things

Hook system call (Global)

When process is retrieving the task information, make it return with host privilege.

Patch code (Global, good for rootkit)

When process is retrieving the task information, make it return with host privilege.
Patch code (Global, good for rootkit)

```
mach_port_name_t
host_self_trap(
        __unused struct host_self_trap_args *args)
{
        ipc_port_t sright;
        mach_port_name_t name;

        sright = ipc_port_copy_send(current_task()->itk_host);
        name = ipc_port_copyout_send(sright, current_space());
        return name;
}
```

```
                                        _host_self_trap:
0xffffff8000225f20 55                   push    rbp
0xffffff8000225f21 4889E5               mov     rbp, rsp
0xffffff8000225f24 65488B042508000000   mov     rax, qword [gs:0x8]
0xffffff8000225f2d 488B8058030000       mov     rax, qword [ds:rax+0x358]
0xffffff8000225f34 488BB820020000       mov     rdi, qword [ds:rax+0x220]
0xffffff8000225f3b E89034FFFF           call    _ipc_port_copy_send
0xffffff8000225f40 65488B0C2508000000   mov     rcx, qword [gs:0x8]
0xffffff8000225f49 488B8958030000       mov     rcx, qword [ds:rcx+0x358]
0xffffff8000225f50 488BB168020000       mov     rsi, qword [ds:rcx+0x268]
0xffffff8000225f57 4889C7               mov     rdi, rax
0xffffff8000225f5a 5D                   pop     rbp
0xffffff8000225f5b E9E034FFFF           jmp     _ipc_port_copyout_send
```

call _host_self
mov rax, [rax+0x20]
mov rdi, rax

Direct Kernel Task Access

Since Mac OS X 10.6, it restricted task access for kernel task

> "task_for_pid() is not supported on the kernel task, no matter your privilege level nor what API you use.
>
> … there is no legitimate use for inspecting kernel memory."

Direct Task Access

We don’t use task_for_pid()

processor_set_tasks(p_default_set_control, &task_list, &task_count)

task_list[0] is the kernel task

We can control all of tasks and read / write memory, even use thread_set_state() to inject dynamic libraries.

Bypass Kernel Module Verification in 10.9

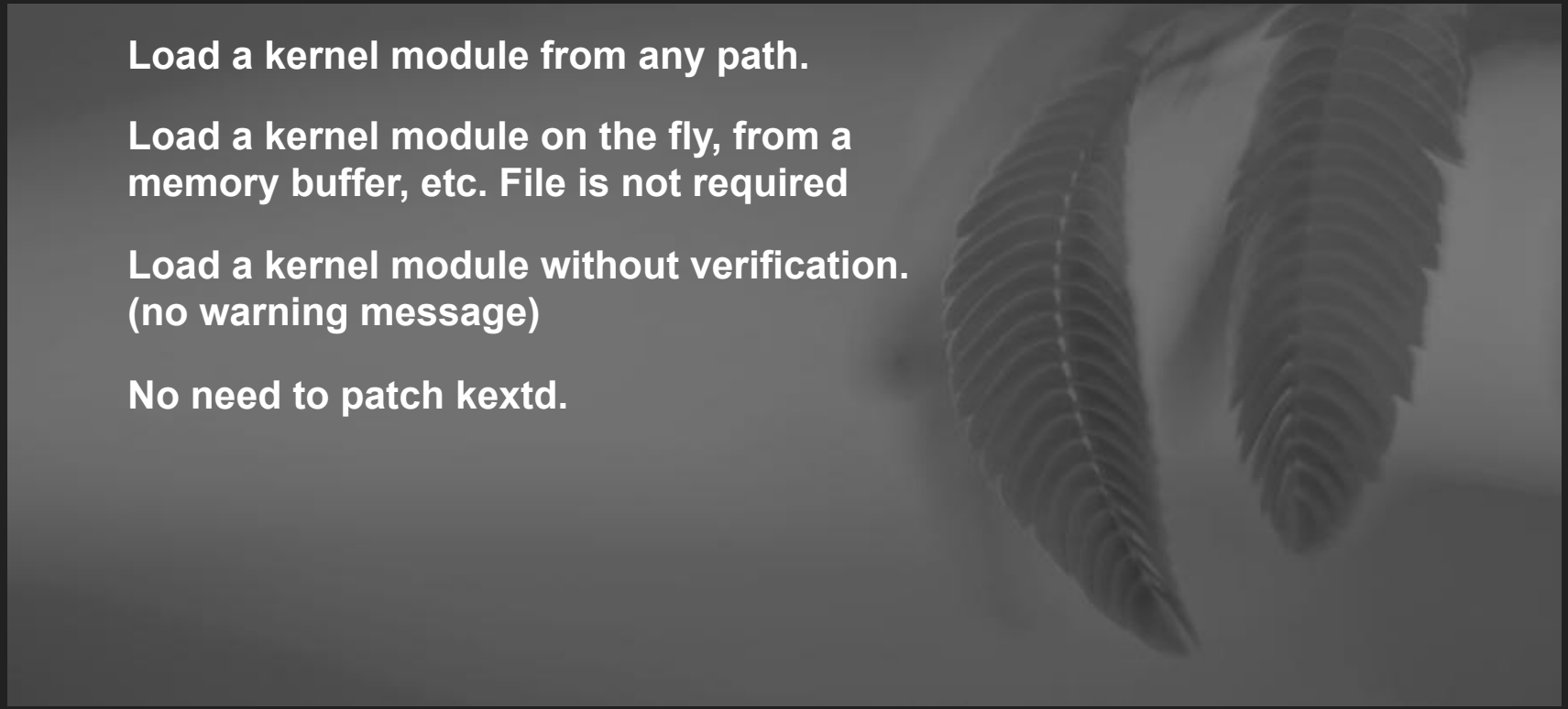
In Mac OS 10.9, if you want to load a kernel module

Put the kernel module file into /System/Library/Extensions/

Run kextload to load the file

If the kernel module is not signed, OS will pop up a warning message

mykextload



Load a kernel module from any path.

Load a kernel module on the fly, from a memory buffer, etc. File is not required

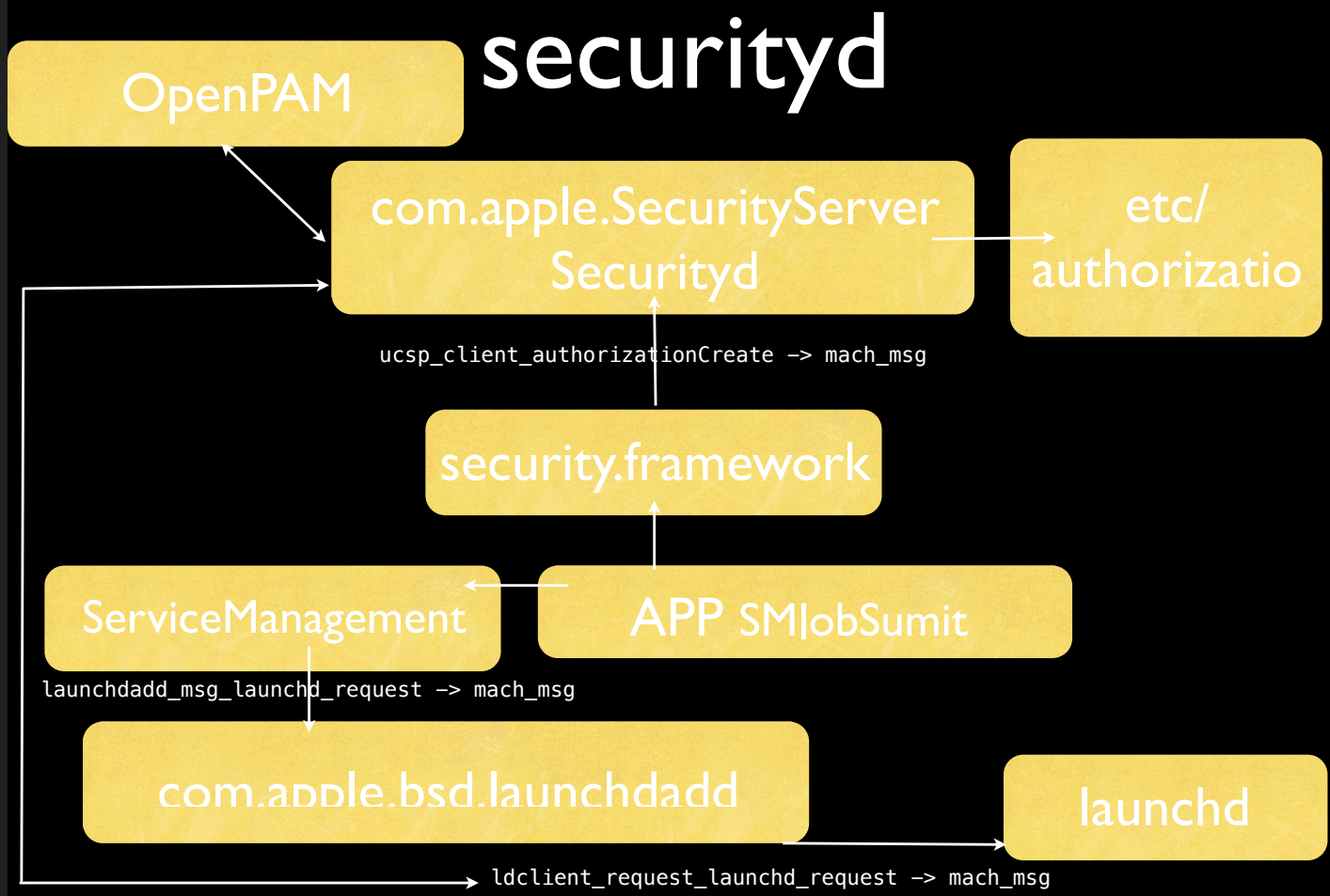
Load a kernel module without verification. (no warning message)

No need to patch kextd.

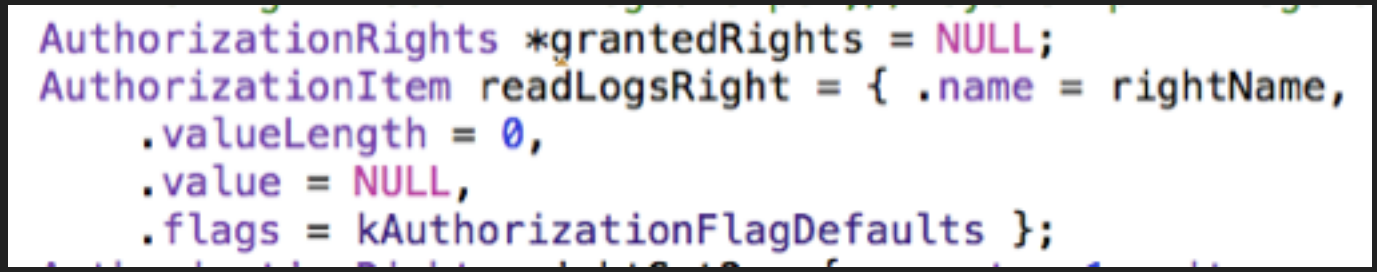
kext_request()

```
typedef struct mkext2_file_entry {
    uint32_t  compressed_size;  // if zero, file is not compressed
    uint32_t  full_size;        // full size of data w/o this struct
    uint8_t   data[0];          // data is inline to this struct
} mkext2_file_entry;

typedef struct mkext2_header {
    MKEXT_HEADER_CORE
    uint32_t plist_offset;
    uint32_t plist_compressed_size;
    uint32_t plist_full_size;
} mkext2_header;
```

A Trick to Gain Root Permission

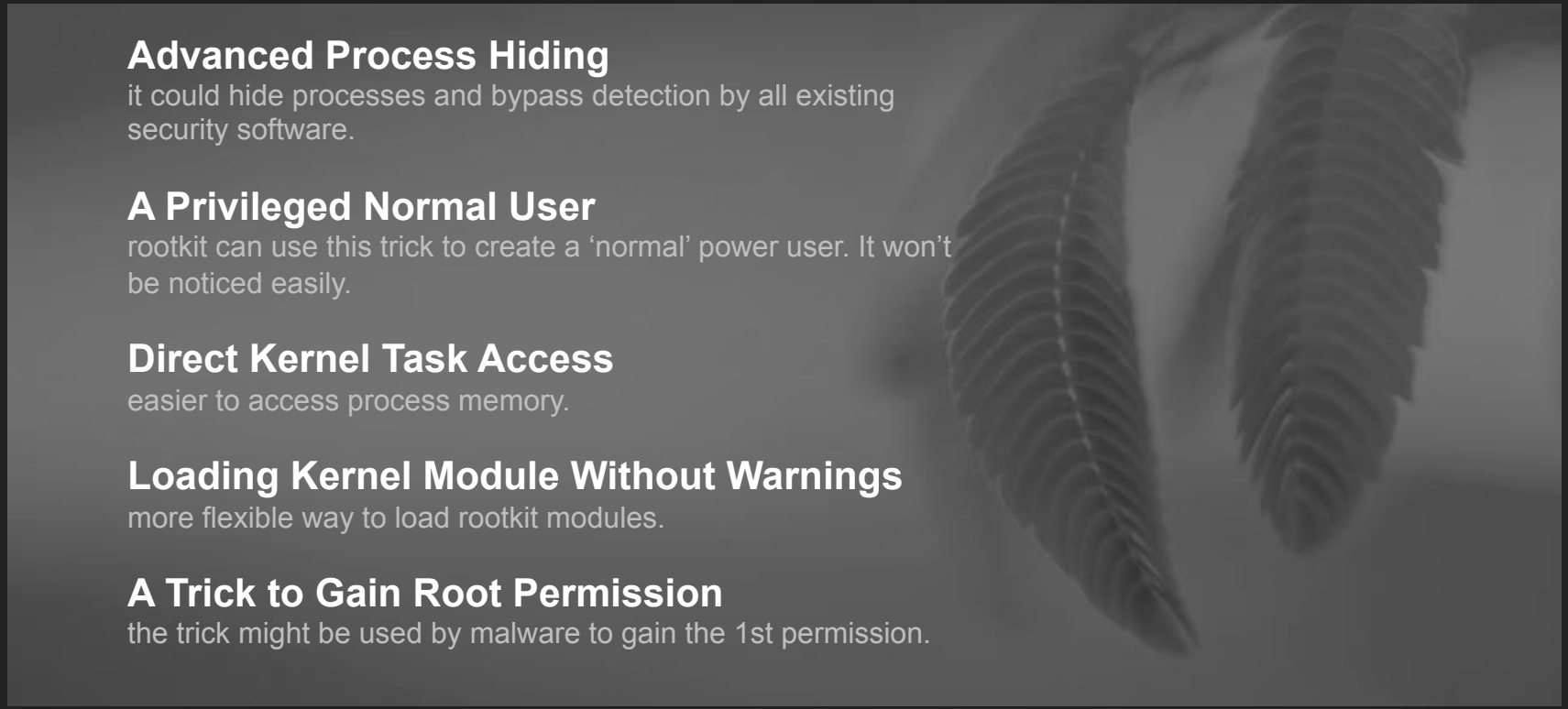

securityd
OpenPAM
com.apple.SecurityServer
Securityd
etc/
authorizatio
ucsp_client_authorizationCreate -> mach_msg
security.framework
ServiceManagement
APP SMJobSumit
launchdadd_msg_launchd_request -> mach_msg
com.apple.bsd.launchdadd
launchd
ldclient_request_launchd_request -> mach_msg

system.privilege.admin
system.privilege.taskport
com.apple.ServiceManagement.daemons.modify
com.apple.ServiceManagement.blesshelper



com.apple.SoftwareUpdate.scan

Conclusion



Advanced Process Hiding
it could hide processes and bypass detection by all existing security software.

A Privileged Normal User
rootkit can use this trick to create a ‘normal’ power user. It won’t be noticed easily.

Direct Kernel Task Access
easier to access process memory.

Loading Kernel Module Without Warnings
more flexible way to load rootkit modules.

A Trick to Gain Root Permission
the trick might be used by malware to gain the 1st permission.


Contact: tt@teamt5.org / ttsecurity@gmail.com